NAME

intro – introduction to games

DESCRIPTION

This section describes the recreational and educational programs found in the directory **/usr/games**. The availability of these programs may vary from system to system.

ARITHMETIC(6)

NAME

arithmetic − provide drill in number facts

SYNOPSIS

/usr/games/arithmetic [+−x/] [range]

DESCRIPTION

Arithmetic types out simple arithmetic problems, and waits for an answer to be typed in. If the answer is correct, it types back "Right!", and a new problem. If the answer is wrong, it replies "What?", and waits for another answer. Every twenty problems, it publishes statistics on correctness and the time required to answer.

To quit the program, type an interrupt (delete).

The first optional argument determines the kind of problem to be generated; **+**, **−**, **x**, and **/** respectively cause addition, subtraction, multiplication, and division problems to be generated. One or more characters can be given; if more than one is given, the different types of problems will be mixed in random order; default is **+−**.

Range is a decimal number; all addends, subtrahends, differences, multiplicands, divisors, and quotients will be less than or equal to the value of *range*. Default *range* is 10.

At the start, all numbers less than or equal to *range* are equally likely to appear. If the respondent makes a mistake, the numbers in the problem which was missed become more likely to reappear.

As a matter of educational philosophy, the program will not give correct answers, since the learner should, in principle, be able to calculate them. Thus the program is intended to provide drill for someone just past the first learning stage, not to teach number facts *de novo*. For almost all users, the relevant statistic should be time per problem, not percent correct.

NAME

back – the game of backgammon

SYNOPSIS

/usr/games/back

DESCRIPTION

Back is a program which provides a partner for the game of backgammon. It is designed to play at three different levels of skill, one of which you must select. In addition to selecting the opponent's level, you may also indicate that you would like to roll your own dice during your turns (for the superstitious players). You will also be given the opportunity to move first. The practice of each player rolling one die for the first move is not incorporated.

The points are numbered 1–24, with 1 being white's extreme inner table, 24 being brown's inner table, 0 being the bar for removed white pieces and 25 the bar for brown. For details on how moves are expressed, type **y** when *back* asks "Instructions?" at the beginning of the game. When *back* first asks "Move?", type **?** to see a list of move options other than entering your numerical move.

When the game is finished, *back* will ask you if you want the log. If you respond with **y**, *back* will attempt to append to or create a file **back.log** in the current directory.

FILES

/usr/games/lib/backrules	rules file
/tmp/b*	log temp file
back.log	log file

BUGS

The only level really worth playing is "expert", and it only plays the forward game.
Back will complain loudly if you attempt to make too *many* moves in a turn, but will become very silent if you make too *few*.
Doubling is not implemented.

BJ(6)

NAME

bj – the game of black jack

SYNOPSIS

/usr/games/bj

DESCRIPTION

Bj is a serious attempt at simulating the dealer in the game of black jack (or twenty-one) as might be found in Reno. The following rules apply:

The bet is $2 every hand.

A player "natural" (black jack) pays $3. A dealer natural loses $2. Both dealer and player naturals is a "push" (no money exchange).

If the dealer has an ace up, the player is allowed to make an "insurance" bet against the chance of a dealer natural. If this bet is not taken, play resumes as normal. If the bet is taken, it is a side bet where the player wins $2 if the dealer has a natural and loses $1 if the dealer does not.

If the player is dealt two cards of the same value, he is allowed to "double". He is allowed to play two hands, each with one of these cards. (The bet is doubled also; $2 on each hand.)

If a dealt hand has a total of ten or eleven, the player may "double down". He may double the bet ($2 to $4) and receive exactly one more card on that hand.

Under normal play, the player may "hit" (draw a card) as long as his total is not over twenty-one. If the player "busts" (goes over twenty-one), the dealer wins the bet.

When the player "stands" (decides not to hit), the dealer hits until he attains a total of seventeen or more. If the dealer busts, the player wins the bet.

If both player and dealer stand, the one with the largest total wins. A tie is a push.

The machine deals and keeps score. The following questions will be asked at appropriate times. Each question is answered by y followed by a new-line for "yes", or just new-line for "no".

```
?                (means, "do you want a hit?")
Insurance?
Double down?
```

Every time the deck is shuffled, the dealer so states and the "action" (total bet) and "standing" (total won or lost) is printed. To exit, hit the interrupt key (DEL) and the action and standing will be printed.

NAME

chess – the game of chess

SYNOPSIS

/usr/games/chess

DESCRIPTION

Chess is a computer program that plays class D chess. Moves may be given either in standard (descriptive) notation or in algebraic notation. The symbol + must be placed at the end of a line when the move on that line places the opponent's king in check. The values **o-o** and **o-o-o** specify castling, king side or queen side, respectively.

The user is prompted for a move or command by a *. To play black, type **first** at the onset of the game. To print a copy of the board in play, type a carriage return only. Each move is echoed in the appropriate notation, followed by the program's reply. Near the middle and end games, the program can take considerable time in computing its moves.

A **?** or **help** may be typed to get a help message that briefly describes the possible commands.

DIAGNOSTICS

The most cryptic diagnostic is "eh?" which means that the input was syntactically incorrect.

BUGS

Pawns may be promoted only to queens.

CRAPS(6)

NAME

craps – the game of craps

SYNOPSIS

/usr/games/craps

DESCRIPTION

Craps is a form of the game of craps that is played in Las Vegas. The program simulates the *roller*, while the user (the *player*) places bets. The player may choose, at any time, to bet with the roller or with the *House*. A bet of a negative amount is taken as a bet with the House, any other bet is a bet with the roller.

The player starts off with a "bankroll" of $2,000.

The program prompts with:

bet?

The bet can be all or part of the player's bankroll. Any bet over the total bankroll is rejected and the program prompts with **bet?** until a proper bet is made.

Once the bet is accepted, the roller throws the dice. The following rules apply (the player wins or loses depending on whether the bet is placed with the roller or with the House; the odds are even). The *first* roll is the roll immediately following a bet:

1. On the first roll:

7 or 11	wins for the roller;
2, 3, or 12	wins for the House;
any other number	is the *point*, roll again (Rule 2 applies).

2. On subsequent rolls:

point	roller wins;
7	House wins;
any other number	roll again.

If a player loses the entire bankroll, the House will offer to lend the player an additional $2,000. The program will prompt:

marker?

A **yes** (or **y**) consummates the loan. Any other reply terminates the game.

If a player owes the House money, the House reminds the player, before a bet is placed, how many markers are outstanding.

If, at any time, the bankroll of a player who has outstanding markers exceeds $2,000, the House asks:

Repay marker?

A reply of **yes** (or **y**) indicates the player's willingness to repay the loan. If only 1 marker is outstanding, it is immediately repaid. However, if more than 1 marker are outstanding, the House asks:

How many?

markers the player would like to repay. If an invalid number is entered (or just a carriage return), an appropriate message is printed and the program will prompt with **How many?** until a valid number is entered.

If a player accumulates 10 markers (a total of $20,000 borrowed from the House), the program informs the player of the situation and exits.

Should the bankroll of a player who has outstanding markers exceed $50,000, the *total* amount of money borrowed will be *automatically* repaid to the House.

Any player who accumulates $100,000 or more breaks the bank. The program then prompts:

New game?

to give the House a chance to win back its money.

Any reply other than **yes** is considered to be a **no** (except in the case of **bet?** or **How many?**). To exit, send an interrupt (break), DEL, or control-D. The program will indicate whether the player won, lost, or broke even.

MISCELLANEOUS

The random number generator for the die numbers uses the seconds from the time of day. Depending on system usage, these numbers, at times, may seem strange but occurrences of this type in a real dice situation are not uncommon.

HANGMAN(6)

NAME

hangman – guess the word

SYNOPSIS

/usr/games/hangman [arg]

DESCRIPTION

Hangman chooses a word at least seven letters long from a dictionary. The user is to guess letters one at a time.

The optional argument *arg* names an alternate dictionary.

FILES

/usr/lib/w2006

BUGS

Hyphenated compounds are run together.

NAME

jotto – secret word game

SYNOPSIS

/usr/games/jotto [–p]

DESCRIPTION

Jotto is a word guessing game. You try to guess the computer's secret word before it guesses yours. Clues are obtained by entering probe words. For example, if the computer's secret word is "brown" and you probe with "stare", it will reply "1" indicating that there is one letter in common between your probe and the secret word. Double letters count only once unless they appear in both words. For example, if the hidden word is "igloo" and you probe with "broke", the computer will reply "1". But if you probe with "gloom", the computer will respond "4". All secret words and probe words should be non-proper English five-letter words. If the computer guesses your word exactly, please respond with "y". It will then tell you what its secret word was. The **–p** flag instructs the computer to report its progress in guessing your word.

BUGS

The dictionary contains some unusual words and lacks some common ones.

NAME

maze – generate a maze

SYNOPSIS

/usr/games/maze

DESCRIPTION

Maze asks a few questions and then prints a maze.

BUGS

Some mazes (especially small ones) have no solutions.

NAME

moo – guessing game

SYNOPSIS

/usr/games/moo

DESCRIPTION

Moo is a guessing game imported from England. The computer picks a number consisting of four distinct decimal digits. The player guesses four distinct digits being scored on each guess. A "cow" is a correct digit in an incorrect position. A "bull" is a correct digit in a correct position. The game continues until the player guesses the number (a score of four bulls).

NAME

quiz – test your knowledge

SYNOPSIS

/usr/games/quiz [**–i** file] [**–t**] [category1 category2]

DESCRIPTION

Quiz gives associative knowledge tests on various subjects. It asks items chosen from *category1* and expects answers from *category2*, or vice versa. If no categories are specified, *quiz* gives instructions and lists the available categories.

Quiz tells a correct answer whenever you type a bare new-line. At the end of input, upon interrupt, or when questions run out, *quiz* reports a score and terminates.

The **–t** flag specifies "tutorial" mode, where missed questions are repeated later, and material is gradually introduced as you learn.

The **–i** flag causes the named file to be substituted for the default index file. The lines of these files have the syntax:

```
line      = category new-line | category : line
category  = alternate | category | alternate
alternate = empty | alternate primary
primary   = character | [ category ] | option
option    = { category }
```

The first category on each line of an index file names an information file. The remaining categories specify the order and contents of the data in each line of the information file. Information files have the same syntax. Backslash \ is used as with *sh*(1) to quote syntactically significant characters or to insert transparent new-lines into a line. When either a question or its answer is empty, *quiz* will refrain from asking it.

FILES

/usr/games/lib/quiz/index
/usr/games/lib/quiz/*

BUGS

The construct "a|ab" does not work in an information file. Use "a{b}".

NAME

reversi – a game of dramatic reversals

SYNOPSIS

/usr/games/reversi [[**–r**] *file*]

DESCRIPTION

Reversi (also known as "friends", "Chinese friends" and "Othello") is played on an 8 by 8 board using two-sided tokens. Each player takes his turn by placing a token with his side up in an empty square. During the first four turns, players may only place tokens in the four central squares of the board. Subsequently, with each turn, a player *must* capture one or more of his opponent's tokens. He does this by placing one of his tokens such that it and another of his tokens embrace a solid line of his opponent's tokens horizontally, vertically or diagonally. Captured tokens are flipped over and thus can be re-captured. If a player cannot outflank his opponent, he forfeits his turn. The play continues until the board is filled or until no more outflanking is possible.

In this game, your tokens are asterisks (*) and the machine's are at-signs (@). You move by typing in the row and column at which you want to place your token as two digits (1-8), optionally separated by blanks or tabs. You can also type in:

- **c** to continue the game after hitting break (this is only necessary if you interrupt the machine while it is deliberating),
- **g** *n* to start *reversi* playing against itself for the next *n* moves (or until the break key is hit),
- **n** to stop printing the board after each move,
- **o** to start it up again,
- **p** to print the board regardless,
- **q** to quit (without dishonor),
- **s** to print the score, and, as always,
- **!** to escape to the shell. Control-d gets you back.

Reversi also recognizes several commands which are valid only at the start of the game, before any moves have been made. They are:

- **f** to let the machine go first.
- **h** *n* to ask for a handicap of from one to four corner squares. If you're *really* good, you can give the machine a handicap by typing a negative number.
- **l** *n* to set the amount of look-ahead used by the machine in searching for moves. Zero means none at all. Four is the default. Greater than six means you may fall asleep waiting for the machine to move.
- **t** *n* to tell *reversi* that you will only need *n* seconds to consider each move. If you fail to respond in the allotted time, you forfeit your turn.

If *reversi* is given a file name as an argument, it will checkpoint the game, move by move, by dumping the board onto *file*. The **–r** option will cause *reversi* to restart the game from *file* and continue logging.

DIAGNOSTICS

"Illegal!" for an illegal move, and "Huh?" for a move that even the machine cannot understand.

NAME

sky – obtain ephemerides

SYNOPSIS

/usr/games/sky [–l]

DESCRIPTION

Sky predicts the apparent locations of the Sun, the Moon, the planets out to Saturn, stars of magnitude at least 2.5, and certain other celestial objects. *Sky* reads the standard input to obtain a GMT time typed on one line with blanks separating year, month number, day, hour, and minute; if the year is missing the current year is used. If a blank line is typed, the current time is used. The program prints the azimuth, elevation, and magnitude of objects which are above the horizon at the ephemeris location of Murray Hill at the indicated time. The **–l** flag causes it to ask for another location.

Placing a "1" input after the minute entry causes the program to print out the Greenwich Sidereal Time at the indicated moment and to print for each body its topographic right ascension and declination as well as its azimuth and elevation. Also, instead of the magnitude, the semidiameter of the body, in seconds of arc, is reported.

A "2" after the minute entry makes the coordinate system geocentric.

The effects of atmospheric extinction on magnitudes are not included; the brightest magnitudes of variable stars are marked with *.

For all bodies, the program takes into account precession and nutation of the equinox, annual (but not diurnal) aberration, diurnal parallax, and the proper motion of stars. In no case is refraction included.

The program takes into account perturbations of the Earth due to the Moon, Venus, Mars, and Jupiter. The expected accuracies are: for the Sun and other stellar bodies a few tenths of seconds of arc; for the Moon (on which particular care is lavished) likewise a few tenths of seconds. For the Sun, Moon and stars the accuracy is sufficient to predict the circumstances of eclipses and occultations to within a few seconds of time. The planets may be off by several minutes of arc.

There are lots of special options not described here, which do things like substituting named star catalogs, smoothing nutation and aberration to aid generation of mean places of stars, and making conventional adjustments to the Moon to improve eclipse predictions.

For the most accurate use of the program it is necessary to know that it actually runs in Ephemeris time.

SEE ALSO

American Ephemeris and Nautical Almanac, for the appropriate years; also, the *Explanatory Supplement to the American Ephemeris and Nautical Almanac*.

NAME

Super Rogue – exploring The Dungeons of Doom (Version 9)

SYNOPSIS

rogue [savefile]

rogue -s

DESCRIPTION

Super-Rogue is a computer fantasy game with a new twist. The game is CRT oriented with connecting rooms as your maze. You must travel from room to room and level to level fighting off various monsters. There are many objects spread out amoung the various rooms to put in your pack. Picking up gold pieces is the main object of the game.

To get started you really only need to know two commands. The command **?** will give you a list of the available commands. The command **/** will identify the things you see on the screen.

To win the game (as opposed to merely playing to beat other people high scores) you must locate the **Amulet of Yendor** which is somewhere below the **35th** level of the dungeon and get it out. Nobody has achieved this yet and if somebody does, he will probably go down in history as a hero among heros.

When the game ends, either by your death, if you chicken-out, or if you (by some miracle) manage to win, **rogue** will give you a list of the top 10 scorers. The scoring is based upon how much gold you get. There is a **10%** penalty for getting yourself killed.

You can save your game for later play by typing in the letter **S**. To resume the adventure, give the name of the save file as the only argument to **rogue**.

For more detailed directions, read the document **Super Rogue Version 9.0**.

FILES

/usr/games/lib/rogue_score	Score file
rogue.save or .rogue.save	Save file
/usr/games/lib/rogue.memo	Super Rogue Document

BUGS

None found at point of release of this manual page.

NAME

ttt, cubic – tic-tac-toe

SYNOPSIS

/usr/games/ttt
/usr/games/cubic

DESCRIPTION

Ttt is the X and O game popular in the first grade. This is a learning program that never makes the same mistake twice.

Although it learns, it learns slowly. It must lose nearly 80 games to completely know the game.

Cubic plays three-dimensional tic-tac-toe on a 4×4×4 board. Moves are specified as a sequence of three coordinate numbers in the range 1-4.

FILES

/usr/games/ttt.klearning file

BUGS

Cubic does not yet work on the VAX.

NAME

wump – the game of hunt-the-wumpus

SYNOPSIS

/usr/games/wump

DESCRIPTION

Wump plays the game of "Hunt the Wumpus." A Wumpus is a creature that lives in a cave with several rooms connected by tunnels. You wander among the rooms, trying to shoot the Wumpus with an arrow, meanwhile avoiding being eaten by the Wumpus and falling into Bottomless Pits. There are also Super Bats which are likely to pick you up and drop you in some random room.

The program asks various questions which you answer one per line; it will give a more detailed description if you want.

This program is based on one described in *People's Computer Company*, 2, 2 (November 1973).

BUGS

It will never replace Adventure.